Single Responsibility of SOLID in Abap

beyhan_meyrali
Active Contributor

2023 Apr 17 10:25 PM

Hi Abapers,

In this post, I would like to share my thoughts and suggestions on SOLID's most violated principle, Single responsibility.

Before we start with SOLID, I need to mention about MVC. Because they go hand in hand. You can read about applying MVC in this blog post.

Previous to Abap, I had chance to work with different programming languages such as Java and C#. And on those platforms, software development has already evolved to create high quality software. But unfortunately same can not be said for Abap development. And most Abapers do not even pay attention SOLID or MVC pattern and do not understand OOP. And as a result, we end up having low quality code.

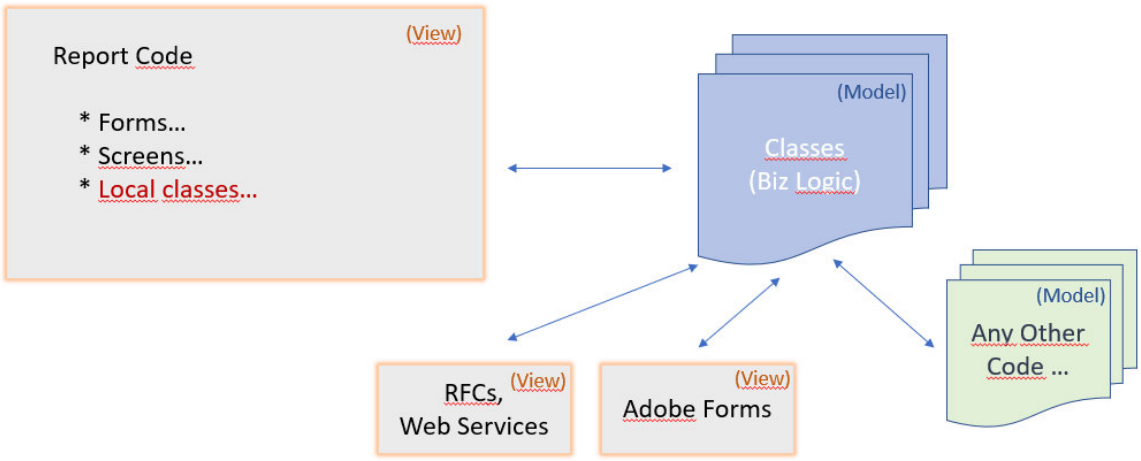
Here are the a few mistakes and my suggestion to create better architecture.

On Local Classes

Use local classes only for view layer logic not for business logic. They should be only responsible for related report's view and command logic. Create a separate class to retrieve, process data in another class. That will comply with MVC, Separation of concerns, Solid's single responsibility.

For example, when a report's data is requested in a RFC, you can call same class object and simply get data from same source. In that way, you make use of same code, instead

of writing another copy of same code and creating possibility of data/process inconsistency. Report belongs to view layer and logic belongs to model layer.



Long methods(Nightmares)

If a method contains too many lines, lets say more than 20 lines, that is because most likely you are doing more than one thing in same method.

Instead, try to think like you are building something with Legos, that you can change parts without need to revive, retest whole code.

Most OOP beginners, create a class and fill all lines in same method. Sorry friends, but, that is hardly OOP. That is just another way of writing function module with many lines of code.

That makes it difficult to test, difficult to maintain. Instead, give meaningful method names, write your code in chunks. Your main methods should call other methods and other methods should encapsulate their complexities in them. Pay attention to visibility scopes and at that point please think about Single responsibility of solid and Lego like coding.

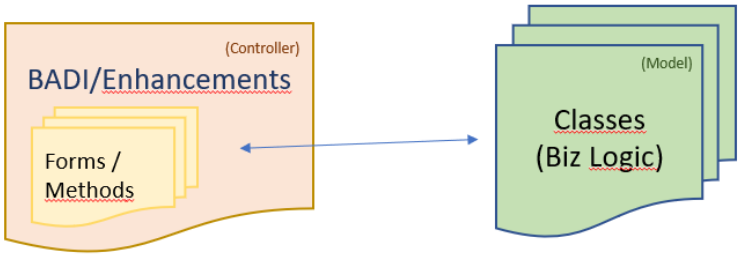
When you have small code blocks with single responsibility, you can maintain them much faster and better. It will be easier to test them. Just think of last time when you go to through a long code to understand usage of a variable. And then imagine if it was a small method with a few lines and how different it would be to understand logic.

On Enhancements and BADI Implementations

Most developers write whole code directly under user-exits, enhancements and Badi methods. Here you need to pay attention single responsibility and MVC.

If that is a multiple use badi, you can create another implementation and write your code in that new class. That is fine as long as you can provide code re-usability. Think of DRY. Because that class is only for your business requirements and somebody else can create another class if they need some other requirement.

But, If that is an enhancement or single use badi, please think of them as controllers of MVC, where you can pass SAP data to your own classes, process required logic and retrieve results. In that way, instead of having many kind of different logic, many lines of codes, in same method or form, you will have calls to methods that encapsulates logic and message related code if you need to show message.



Sample User-Exit implementation. Here user exit used as controller of MVC and all biz logic is processed in instance of class.

```
FORM USEREXIT_SAVE_DOCUMENT_PREPARE.
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
*$*$-Start: (4)---------------------------------------------------------------
ENHANCEMENT 1  ZSD_TAS_MV45AFZZ.    "active version

    DATA lt_vbap TYPE zsd_tas_cl_vaxx_exit_enhcn=>ty_t_xvbap.
    MOVE-CORRESPONDING xvbap[] TO lt_vbap[].

    "Tas Süreci Entegrasyonu
    DATA(tas_obj) = NEW zsd_tas_cl_vaxx_exit_enhcn( ).
    tas_obj->before_saving_doc(
    EXPORTING
      trtyp  = T180-TRTYP
      vbak   = vbak
      xvbap  = lt_vbap
    IMPORTING
      status = DATA(tas_status)
    ).
    IF tas_status-status ne tas_obj->c_stat-success.
      MESSAGE tas_status-status_text TYPE 'E' RAISING ERROR.
    ENDIF.

ENDENHANCEMENT.
```

And finally, if you are not already familiar with other programming languages, I definitely suggest you to learn any other object oriented language, such as C#, and improve your skills with rich resources and apply them in Abap.

Those are my brief suggestions for better Abap code. I hope that blog post helps you and gives you an idea. Feel free to add your suggestions and questions.

Thanks for reading.

Some Links

MSDN - on SOLID

Single Responsibility

Separation of Concerns

Tags:

mvc architecture | seperation of concerns soc | SOLID

Comments

matt Active Contributor

2023 Apr 18
11:56 AM

I have to maintain code that I inherited, which has methods with 300 or more lines of code. I see things like

```
CASE groove.
  WHEN this_funky_thang.
    " Do this funky thang
    ... a billion lines of decidely unfunky code
  WHEN that.
    " Do that funky thang
    ... a billion lines of decidely unfunky code
ENDCASE.
```

I may refactor this to the simpler

```
CASE groove.
  WHEN this.
    do_this_funky_thang( ).
  WHEN that.
    do_that_funky_thang( ).
ENDCASE.
```

Or take that funky thang, which isn't really directly connected to groove into its own class, Thereby solidly putting the **S** in SOLID.

```
data(groove_handler) = groove_handler_factory=>get_instance( groo
groove_handler->do_that_funky_thang( ).
```

Whenever I see a section of code which has a descriptive comment above it about what it does, that indicates to me that the functionality should be in its own method - or a new class is needed.

Of course in the above examples, the type of **funky_thang_handler** should be with reference to something like **IF_GROOVE_HANDLER** thereby solidly putting the **D** in SOLID.

I learned Java so that I could get a grip on OO ABAP.

beyhan_meyrali Active Contributor

2023 Apr 18
12:39 PM

Hi Matthew,

Refactoring can be truly annoying sometimes :). I hope more and more Abapers will apply more SOLID and MVC to their codes.

Thanks for sample code.

Regards

M matt Active Contributor

2023 Apr 18
4:15 PM

Refactoring has to be done. I've encountered people who don't like it, because they say that you shouldn't fix what ain't broke. They've never heard of technical debt...